**Overall Status –**

There was quite a similarity in the insertion and deletion parts of the project, for both have to first seek the record in the B+ tree and find the entry to work upon. This helped us to understand the concept to be applied to implement deletion part of the algorithm. Recursion has to be applied to deletion too, in a manner similar to the one applied for insertion. Thus our code starts at the root node, searches for the element to be deleted in a recursive manner. It reaches the leaf where the element is present – if it is present at all, else it returns null. After deleting the element, it checks for redistribution. If redistribution is possible, the source code makes a redistribution among the siblings using the built in method. But, if redistribution is not possible, we have to merge them, and take care of a few things, e.g. the pointers that become redundant. We have handled merging by implementing our own method. Now, there are two different merging methods. One for the leaf node, which is \_merge\_leaf(). And the other for the index node, which is \_merge\_index().

**File Description –**

It wasn’t necessary to create any new file. Provided files, and built in methods were adequate for the implementation of deletion.

**Division of Labor –**

As for division of labor, we divided the task to be accomplished with fairness. The names of team members are Nabilahmed Patel and Archan Joshi. We divided and understood different aspects of the algorithm and code (both, given by professor and from another external source for a better understanding of the working of this kind of project). Like, for example, Archan understood the insertion part of the code, while Nabilahmed understood the deletion part. The coding part was also divided similarly. We discussed the technical do-how’s between us before incorporating any part of the code. We spent almost similar number of hours on this project. Archan spent almost 25 hours and Nabiahmedl spent 30 hours for the said project.

**Logical Errors –**

* If we call the freePage() method without calling unpinPage() on same page, it throws a PinPageException. We corrected it and first unpinned the page using unpinPage() method and then called the freepage() method.
* We found another logical error when we call getSibling() by currentPage instead of it’s parentPage which gave us a wrong result and then we realized and corrected it.
* We encountered third logical error when we call redistribute() by currentPage instead of siblingPage it gave us a wrong output, after realizing that we corrected it.